

## REMARKS

The Examiner is thanked for her courtesy during a telephonic interview on 17 May 2004. During this interview, potential claim amendments were discussed in light of some of the cited references. Claims 38 and 60 have been amended. Claims 38-72 remain pending.

The Examiner has noted that the Information Disclosure Statement (IDS) fails to comply with 37 CFR 1.98(a)(1), which requires a list of all patent, publications, or other information submitted for consideration by the Office. This IDS has been resubmitted with the required list in compliance with 37 CFR 1.98(a)(1).

The Examiner has rejected claims 38-72 under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particular point out and distinctly claim the subject matter which applicant regards as the invention since the limitation “the particular resource” lacks antecedent basis. Claims 38 and 60 have been amended to correct the antecedent basis of this limitation and it is respectfully submitted that the pending claims meet the requirements of 35 U.S.C. §112, second paragraph.

The Examiner rejected claim 38 under 35 U.S.C. §102(e) as being anticipated by Seguchi et al. (U.S. patent 6,633,898). Claims 38 and 60 are rejected under 35 U.S.C. §102(e) as being anticipated by Jarriel et al. (U.S. patent 6,553,403). Claims 38 and 60 are rejected under 35 U.S.C. §102(e) as being anticipated by Pogue et al. (U.S. patent 6,112,240). Claims 38-46, 50-58, 60-63, 66-69, and 70-71 are also rejected under 35 U.S.C. §103(a) as being unpatentable over Culbert (U.S. patent 5,838,968) in view of Judge (U.S. patent 6,430,570). Additionally, claims 47-49 and 64-65 are rejected under 35 U.S.C. §103(a) as being unpatentable over Culbert in view of Judge and further in view of Mayle et al (US 6,182,022). Claims 59 and 72 are rejected under 35 U.S.C. §103(a) as being unpatentable over Culbert in view of Judge and further in view of Applicant's admitted prior art. The Examiner's rejections are respectfully traversed as follows.

Claim 38 requires “for each code downloaded to the computer system, associating a resource indicator with all threads that are executed directly by the downloaded code and all threads that are initiated by the downloaded code, wherein all of the threads that are executed directly by the downloaded code and all threads that are initiated by the downloaded code are defined as a set of related code.” That is, a resource indicator is associated with all the related threads executed directly or indirectly by a particular downloaded code. Claim 1 also requires “updating the resource indicator when the related code changes its actual collective resource usage of a particular resource so that the resource indicator only tracks actual resource usage of

the related code.” Independent claim 60 is directed towards a computer readable medium having computer code for performing similar operations as claim 38.

In other words, the resource indicator is updated so as to track the actual resource usage of only the related code which is defined as the threads executed or initiated by the particular code downloaded to a computer system. Thus, the resource indicator tracks actual resource usage of only the threads executed or initiated by the downloaded code. This feature would advantageously allow implementation of procedures with respect to each set of threads executed or initiated on behalf of each downloaded code when actual resource usage by such related threads exceeds a particular limit. For example, these related threads can be terminated together when the resource indicator which tracks changes in actual resource usage only of these threads indicates that they are exceeding their resource usage.

In contrast, the cited references merely teach using a resource tracking variable to track resource usage of an entire systems or device, not resource usage by a set of related code executed directly or indirectly by a particular downloaded code, *e.g.*, applet, in the manner claimed.

Primary reference Seguchi (US 6,633,898) merely uses a system monitoring thread to monitor system resources *of the server* or the system resources *of the client*. Although Seguchi does teach the use of a system monitoring thread that is accessible by each service module (Figure 2), Seguchi describes this system monitoring thread throughout the specification as monitoring the server or client system, rather than monitoring particular sets of related code. In general, Seguchi is directed towards service modules which execute on the server and/or client based on resource shortages on the server and/or client. See Abstract. At Col. 9, Lines 29-38, Seguchi states that “a system monitoring thread STH1 which continuously manages the processing capability of and a work load *to the server*.” (Emphasis Added). Seguchi also teaches that this “system monitoring thread STH1 monitors the *system* of the server at a specified time interval.” (Emphasis Added). Although Seguchi teaches a server data management table for monitoring information regarding each service module, this table does not list any information regarding a particular service module’s resource usage. See Figure 3 and Col. 9, Lines 45 through Col. 10, Line 3. At Col. 10, Lines 15-24, Seguchi discloses a similar system monitoring thread, but for monitoring the client *system*.

In Seguchi, this system monitoring thread can be used to check the work load of a client or server when a service module is to be loaded onto such server or client. That is, a service module may not be loaded onto a client that that is overloaded. See Col. 14, Lines 45-66. However, it is the resource usage of the entire system (*e.g.*, client) that is checked by the

monitoring thread, not the resource usage of a particular set of related code in the manner claimed. See Col. 15, Lines 6-17: “functions, processing capability of and work load *to the current client* are checked by means of a system call to the operating system COS. This *system* check of the client can also be made by the system monitoring thread CTH1 described above.” (Emphasis Added). Additionally, Seguchi states that “the client daemon CD1 starts only the common modules, which the client daemon CD1 can execute, of those to be executed according to information on *system* resources in the client acquired by the system monitoring thread CTH1.” See Col. 16, Lines 31-35, Emphasis Added. Seguchi also teaches a “description is made for the processing for shifting to an environment for execution of a common module in the server when processing capability of or an excessive workload *to a system* is detected by the system monitoring thread CTH1.” See Col. 16, Lines 49-53. (Emphasis Added). In this context, the passage cited by the Examiner (Col. 23, Lines 33-41) disclose executing a service module in the client only when the system monitoring thread CTH1 does not detect excessive resource usage in the *client*. In sum, Seguchi fails to teach or suggest a resource indicator for tracking changes in actual resource usage *only by a particular set of related code* executed directly or initiated by a downloaded code in the manner claimed.

Primary reference Jarriel discloses a distributed computer environment that is managed in a distributed manner. This managed environment is divided into managed regions which are each managed with its own management server. See Fig. 1 and Col. 4, Lines 6-24. Each management server serves a number of Gateways, and each Gateway has a management framework that facilitates execution of tasks to manage resources in the managed region (MR). See Col. 4, Lines 25-26 and 49-51. The passage cited by the Examiner (Col. 13, Lines 33-41) discloses that this resource monitoring capability can be applied to any subset of *endpoints* or rather systems. Jarriel also discloses that resources are monitored generally in the runtime environment and these monitored resources can include memory, systems, programs, etc. (Col. 7, Lines 7-25) These resources are inherently used by other entities whose resource usage is not differentiated or tracked. In other words, Jarriel is disclosing the monitoring of resources, not using a resource indicator to track changes in actual resource usage *only by a particular set of related code* executed directly or initiated by a downloaded code in the manner claimed.

Primary reference Pogue discloses tracking client information each time a user accesses a web page. See Abstract and Col. 2, Lines 38-46. That is, client information is gathered each time a web page is accessed. See Col. 4, Lines 16-18. Although Pogue can be said to teach that client information is stored each time a resource (*e.g.*, a web page) is being used (*e.g.*, the web page is accessed) by a browser (*e.g.*, resource user), Pogue does not describe using a resource indicator to track the changes in resource usage by such resource user. Thus, Pogue fails to teach or disclose using a resource indicator to track changes in actual resource usage *only by a*

*particular set of related code* executed directly or initiated by a downloaded code in the manner claimed.

Upon a more careful review, Culbert does not teach or suggest tracking of resource usage by each task. In contrast, Culbert describes a mechanism for each task to specify its *required* or *desired* resource usage. See Column 8, Lines 25-59, Emphasis Added: “task 350 has 3 task resource utilization records, 310, 320, and 330. Record 310 contains a list of resources that task 350 *would like* to have available while executing. If task 350 cannot be allocated the resources specified in 310, task 350 could execute and perform its functions with the resources specified in record 320 or 330.” In other words, each task indicates *desired* resources, and not *actual* resource usage by the task. A resource manager 170 “is responsible for creating and dynamically managing the resources available to tasks.” See Col. 6, Lines 45-47. This resource manager 170 is also “responsible for *globally* maximizing resource utilization across *all* XOS tasks.” See Col. 6, Lines 52-54, Emphasis Added. To accomplish this goal, the resource manager 170 maintains current allocation information for each resource in a master list 200, which holds the state of all resources. See Col. 6, Lines 54-63. Although this master list is updated to reflect *global system* resource usage (see Col. 7, Lines 20-27), this master list is not updated to track changes in actual resource usage *only by a particular set of related code* executed directly or initiated by a downloaded code in the manner claimed.

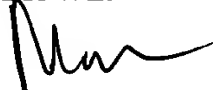
The secondary references also fail to teach or suggest using a resource indicator to track changes in actual resource usage *only by a particular set of related code* executed directly or initiated by a downloaded code in the manner claimed. For example, Judge generally describes an application manager for use in a Java system. See Col. 3, Lines 9-15. The application manager is “responsible for the downloading, execution, and caching of other Java based programs.” See Col. 3, Lines 52-55. The application manager also appears to manage memory by setting limits for the amount of free memory. When the limit is reached, an application is unloaded. See Col. 5, Lines 3-7. Queries may also be made regarding the amount of total memory of the Java environment memory. See Col. 4, Lines 65-67. The memory amount is tracked with respect to overall usage, rather than with respect to what entity is using such memory. Specifically, Judge discloses “the JVM running out of memory.” See Col. 8, Line 2. Judge continues to describe monitoring the free memory level in general so as to trigger the unloading of applications. See Col. 8, Lines 44-47.

For the foregoing reasons, it is respectfully submitted that claims 38 and 60 are patentable over the cited references.

The Examiner's rejections of the dependent claims are also respectfully traversed. However, to expedite prosecution, all of these claims will not be argued separately. Claims 39-69 and each depend directly from independent claims 38 or 70 and, therefore, are respectfully submitted to be patentable over cited art for at least the reasons set forth above with respect to claims 38 and 60. Further, the dependent claims require additional elements that when considered in context of the claimed inventions further patentably distinguish the invention from the cited art. For example, claims 57 and 70 further require that "determining which threads are to be defined as the set of related code based on which threads are assigned to a same protection domain." Claims 58 and 71 further require "aborting the threads of the related code when their resource indicator exceeds a maximum level." Finally, claims 59 and 72 require that "the computer system is integrated with a set top box or a navigational system." The cited references fail to teach or suggest such limitations.

Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,  
BEYER WEAVER & THOMAS, LLP



Mary Ramos Olynick  
Reg. 42,963

P.O. Box 778  
Berkeley, CA 94704-0778  
(510) 843-6200